



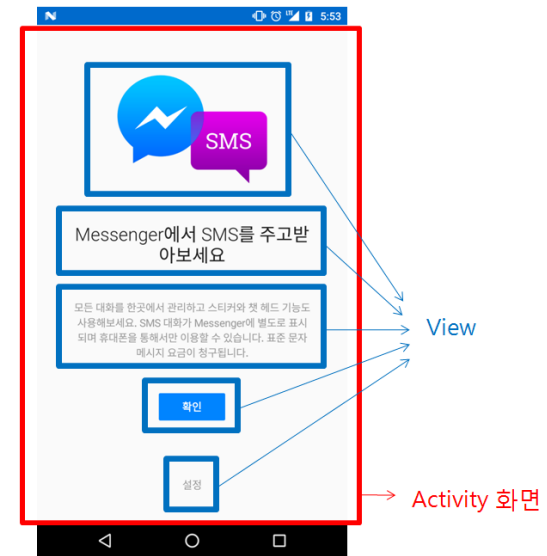
21장. 안드로이드 UI 프로그래밍



21.1 기본적인 UI 프로그래밍 방법

21.1.1. 액티비티-뷰 구조

- 액티비티 자체는 앱의 실행 단위인 컴포넌트
- View는 버튼, 문자열, 이미지 등의 출력 내용
- setContentView(view: View)
- setContentView(layoutResID: Int)



21.1 기본적인 UI 프로그래밍 방법

21.1.2. UI 프로그램 작성 방법 - 코틀린 코드 VS 레이아웃 XML

코틀린 코드로 화면 구성

- 레이아웃 XML 파일 자체를 만들지 않고 코틀린 코드로 모든 뷰를 직접 생성하며, 메서드를 이용하여 뷰 설정을 일일이 지정

```
val linearLayout = LinearLayout(this)
```

```
val button1 = Button(this)  
button1.text = "Button1"  
linearLayout.addView(button1)
```

```
val button2 = Button(this)  
button2.text = "Button2"  
linearLayout.addView(button2)
```

```
setContentView(linearLayout)
```

21.1 기본적인 UI 프로그래밍 방법

레이아웃 XML로 화면 구성

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 1" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 2" />
</LinearLayout>
```

21.2 뷰의 기초 속성

21.2.1. id 속성

- i뷰의 식별자 속성, 필수 속성은 아니며 필요할 때 추가
- 지정한 id 값은 R.java 파일에 등록
- XML에서 등록한 id 값을 매개변수로 하여 findViewById() 함수로 획득

```
<TextView
    android:id="@+id/myText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="hello"/>
```

```
public static final class id {
    //.....
    public static final int myText=0x7f0b0059;
    //.....
}
```

```
val myTextView1: TextView = findViewById(R.id.myText)
```

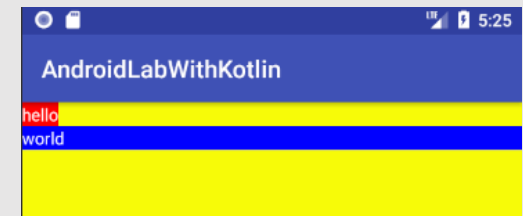
```
val myTextView2 = findViewById<View>(R.id.myText) as TextView
```

21.2 뷰의 기초 속성

20.2.2. layout_width, layout_height 속성

- 뷰의 크기를 지정하기 위한 속성
- match_parent, fill_parent, wrap_content, 100px
- match_parent와 fill_parent는 의미상 동일, 뷰의 크기를 부모 계층의 뷰가 지정한 크기에 꼭 들어차게 자동으로 결정
- wrap_content는 해당 뷰의 내용을 화면에 보이기 위한 적절한 크기를 계산해서 결정

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#F7FB08">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="hello"
        android:background="#FF0000"
        android:textColor="#FFFFFF"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="world"
        android:background="#0000FF"
        android:textColor="#FFFFFF"/>
</LinearLayout>
```



21.2 뷰의 기초 속성

21.2.3. visibility 속성

- 기본값은 “true”이며, “invisible”, “gone” 으로 지정하여 화면에 안 보이게 함



```
if(p0 == trueBtn){  
    targetBtn.visibility = View.VISIBLE  
}else if(p0 == falseBtn){  
    targetBtn.visibility = View.INVISIBLE  
}
```

21.3. 조금 더 코틀린답게 – Kotlin Android Extension

- findViewById() 함수를 이용하지 않고 쉽게 레이아웃 XML 파일에 등록된 View 객체 획득
- 자바로 안드로이드 앱을 개발할 때 butterknife등의 라이브러리를 이용
- Kotlin Android Extension은 확장 플러그인 형태로 제공

```
apply plugin: 'com.android.application'  
apply plugin: 'kotlin-android'  
apply plugin: 'kotlin-android-extensions'
```

- `import kotlinx.android.synthetic.main.activity_main.*`

21.3. 조금 더 코틀린답게 – Kotlin Android Extension

```
import kotlinx.android.synthetic.main.activity_lab1_4.*

class Lab1_4Activity : AppCompatActivity(), View.OnClickListener {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_lab1_4)

        btnVisibleTrue.setOnClickListener(this)
        btnVisibleFalse.setOnClickListener(this)
    }

    override fun onClick(p0: View?) {
        if(p0==btnVisibleTrue){
            textVisibleTarget?.visibility = View.VISIBLE
        }else if(p0 == btnVisibleFalse){
            textVisibleTarget?.visibility = View.INVISIBLE
        }
    }
}
```